NetScript

Y. Yemini (YY)
Distributed Computing & Communications (DCC) Lab
Columbia University; http://www.cs.columbia.edu/dcc

Projects & Participants

- NetScript: a language system to program ANet
- ActiWare: mddlware for end-end mgmt of ANet
  - Virtual Active Nets (VAN)
  - NESTOR:automating config mgmt
- Applications:
  - ASN: active sensor networks
  - Active global fencing
  - Active protocol-based simulations

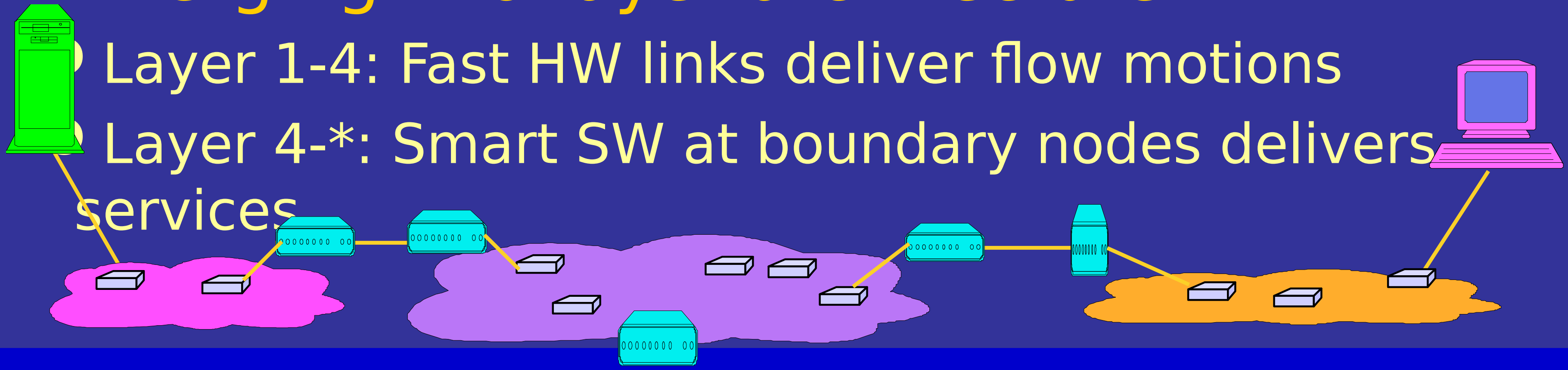
- Columbia CS:
  - Y. Yemini, D. Florissi, H. Schulzrinn P. Wang
  - S. Dasilva, G. Su, A. Konstantinou
  - H. Huang +++
- Columbia Lamont-Doherty:
  - W. Menke ++
- John Hopkins:
  - B. Awerbuch, Y. Amir +++

Overview

- Background
- NetScript
- Applications studies:
  - Active sensor nets
  - Active global protection fences
  - Active protocol-based simulations

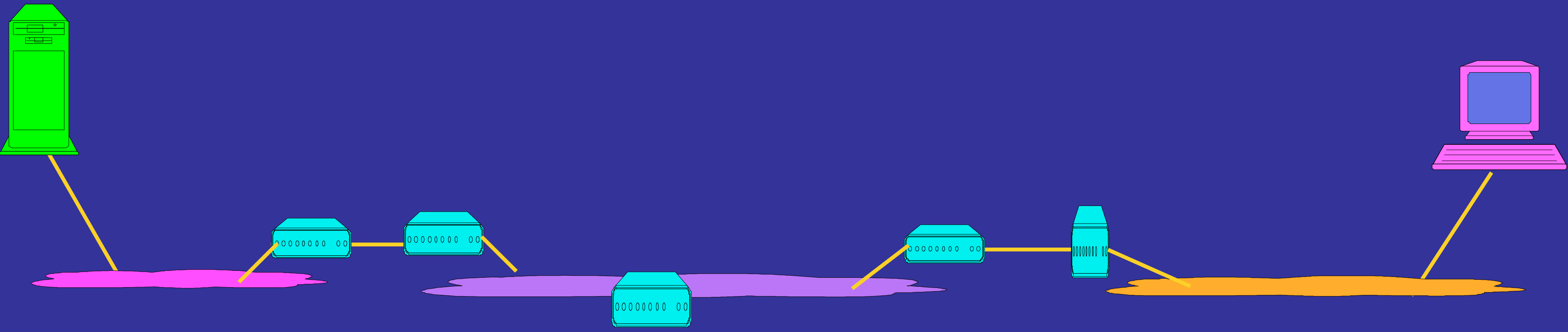
Background Trends

- Layers 3 and below are hardwarized for speed
- Applctn-layer services are distributed in the net
  - E.g., load distribution, caching, filtering, qos, acctng...
  - Disappearing boundaries between end/intermediate-node
- Emerging two-layer architecture:
  - Layer 1-4: Fast HW links deliver flow motions
  - Layer 4-*: Smart SW at boundary nodes delivers services



ANet: Architecture For Net SW

- ANets enable programmable open boundary GW
  - Simplify development & applications of net SW
  - Create a market for net SW and smart services
- A paradigm to program and deploy net SW
- Enable significant new high-layer smart in nets



Base Challenges

- How to program active networks
- How to manage active nets
- How to protect active nets
- ?What significant new capabilities/apps/services will active nets enable?

NetScript

What Are Active Nets Good For?

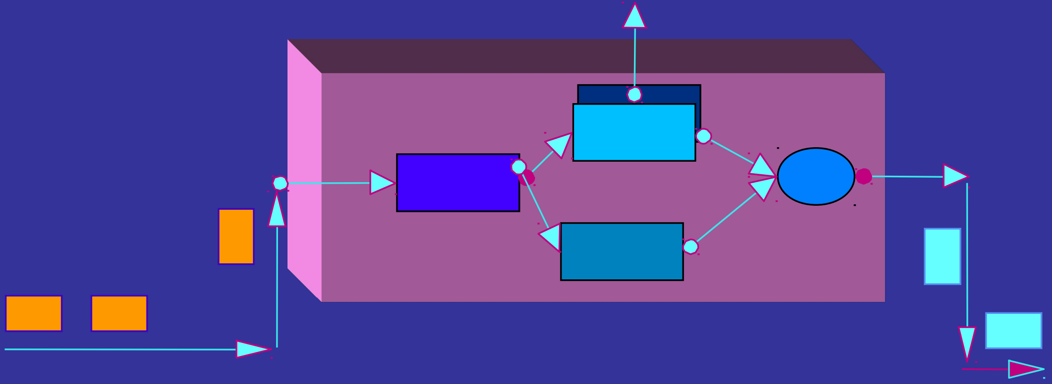
- ☛Active protocols: Multicast/multimedia protocls, signalling protocls
- ☛Active network mgmt: Active monitoring, analysis & config mgmt
- ☛Active security: Active firewall filters & proxies, intrusn detectrs
- ☛Active app layer: Application layer routers, caching servers, filtering/compression/coding, active phone/video

A Common Computational Model

- Packet stream processing
- Synthesizing end-end behaviors by composing local components

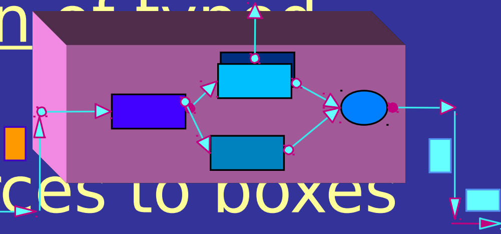
A Language-Based Approach to ANet

- Challenge: how to program active networks
  - Program= compose & coordinate packet-flow processing
- Approach: postscript as a blueprint
  - Language abstrctns to compose progrmmbl pckt-flow procssrs
  - Program networks --end-end services -- not just nodes
- Why a new language?



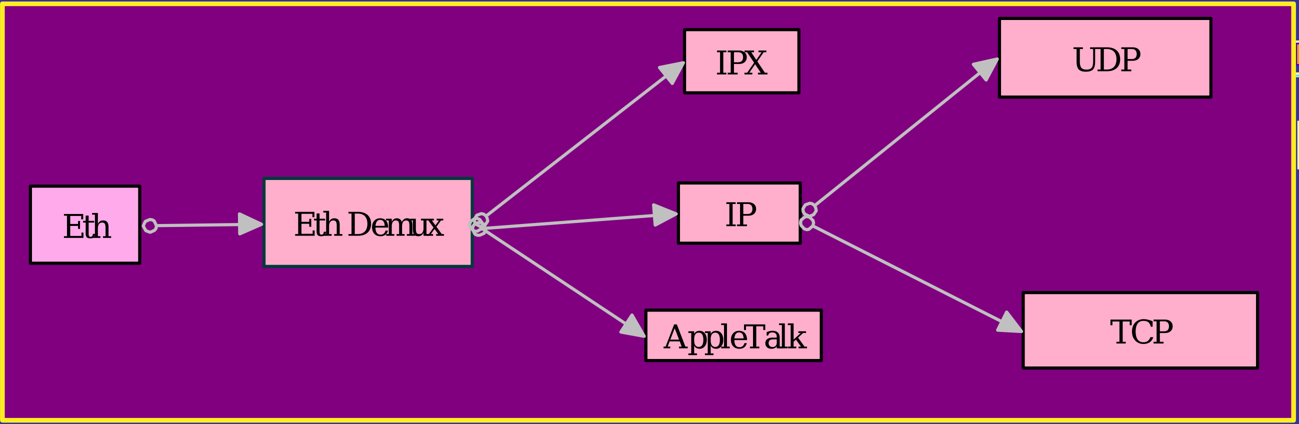
The NetScript Language

- Dataflow model: reactive packet-flow processing
  - active element = packet-flow processor engine
- Dynamic composition of active elements
  - Box is the central construct; represents a flow operator
  - Dynamic composition by interconnection of typed ports

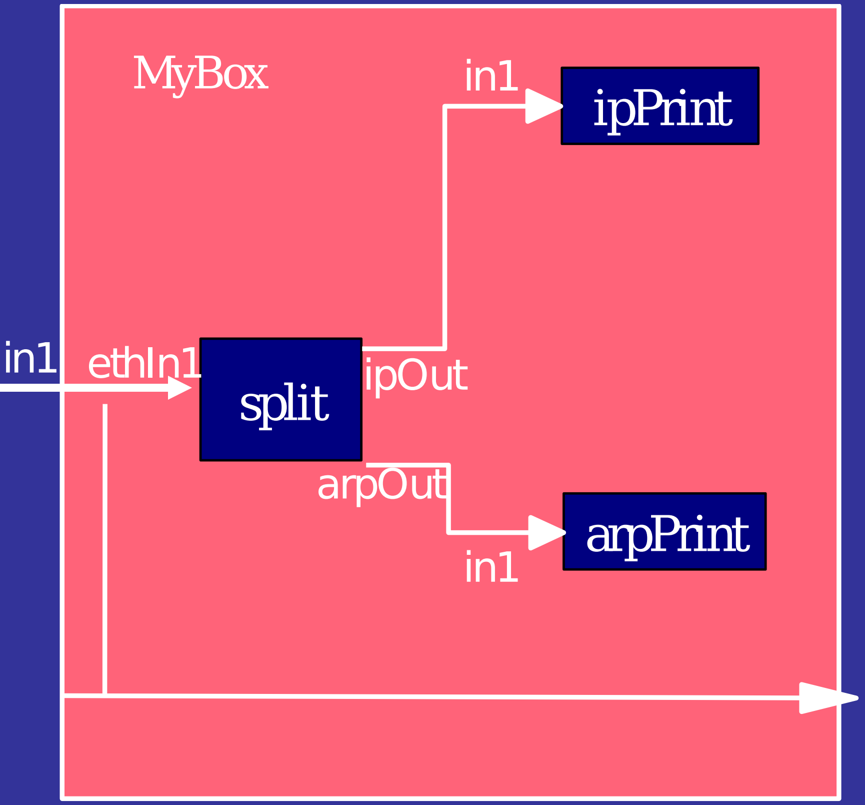


sources to boxes

bridging



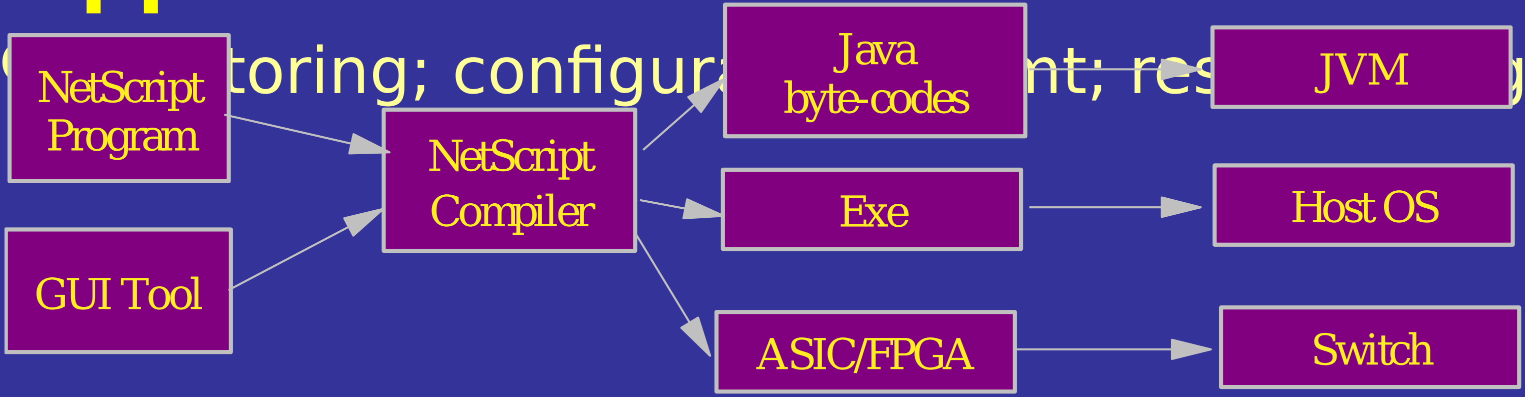
Example: Box Composition



```
box MyBox
{
  inport void EthIn (Eth pkt);
  outport void EthOut (Eth pkt);

  EthIn in1;
  EthOut out1;

  EthSplitter split;
  IPPrinter ipPrint;
  ArpPrinter arpPrint;

  connect
  {
    in1 -> split.ethIn;
    split.arpOut ->arpPrint.in1;
    split.ipOut -> ipPrint.in1;
    in1 -> out1;
  }
}
```

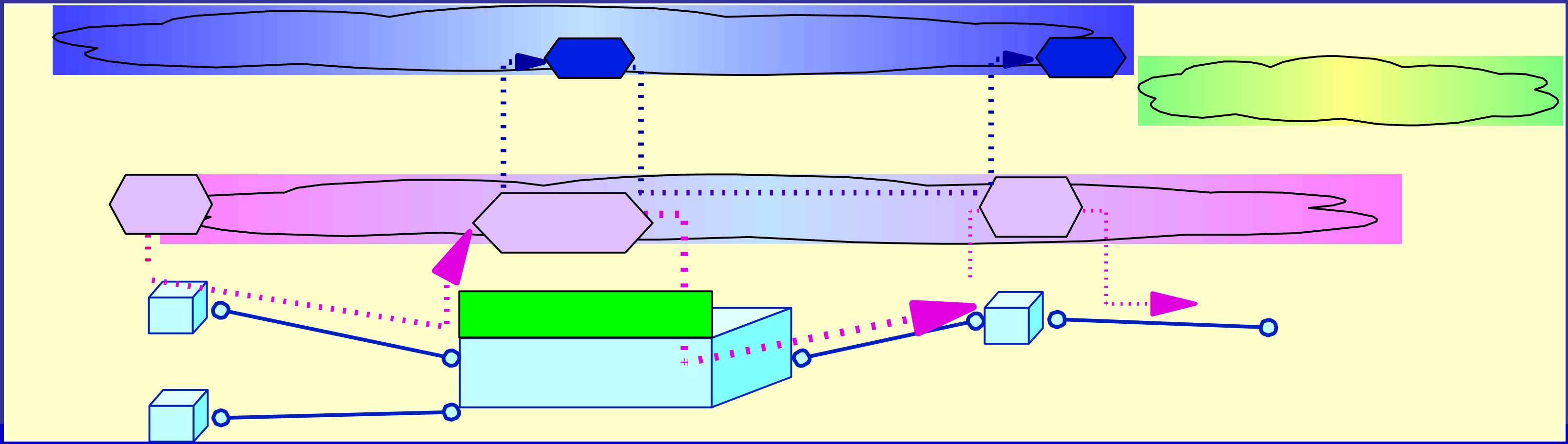
Architecture

- **Language Components**
  - Dataflow composition; pkt presentation; pkt classification
- **Multiple Target Node Architectures:**
  - Java byte codes; Binary executables; ASICs, FPGAs
- **Compile-time generation of mgmt support**
  - toring; configur nt; res gmt...



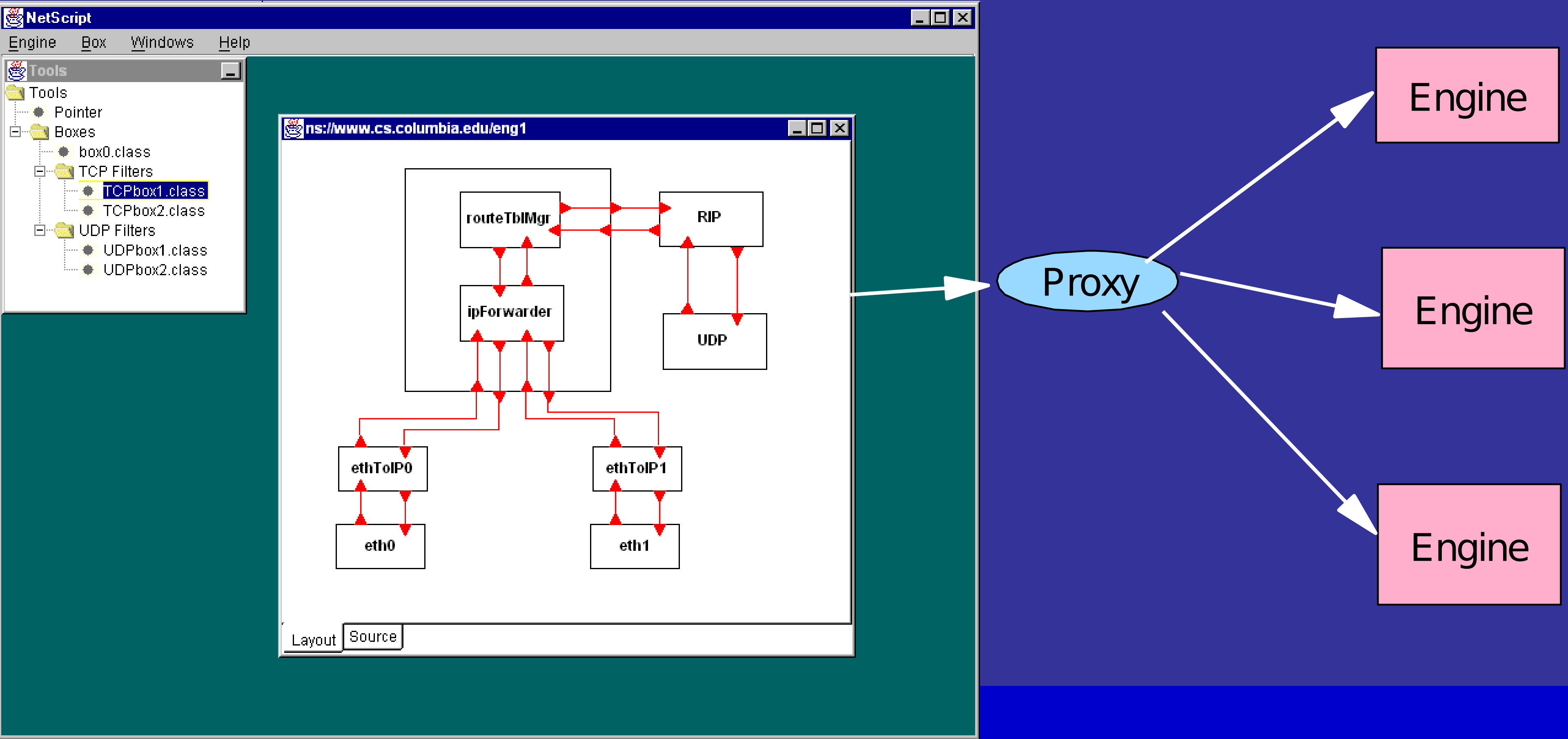
Virtual Active Networks (VAN)

- How to deploy, manage & protect large ANets?
- VAN is a composable unit of end-end service
  - Composition through interconnection, layering and bridging
- VAN is a unit of coordinated resource mgmt
- VAN is a unit of protection



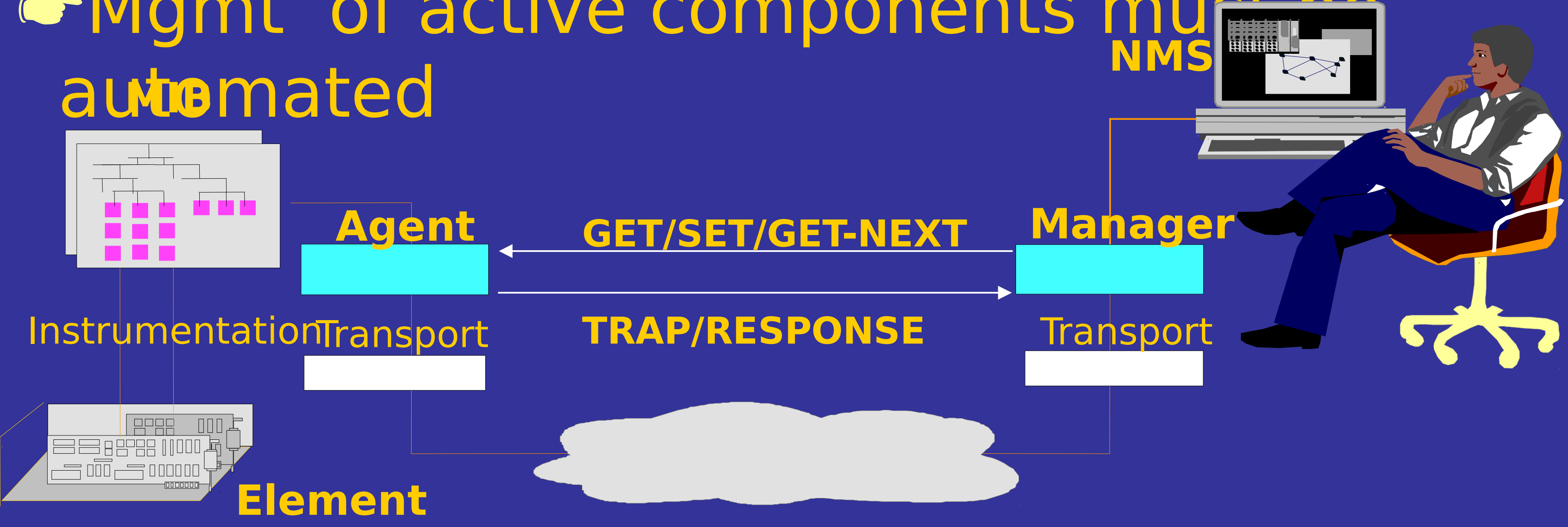
Distributed Run-Time Mgmt

- Delegation Mechanisms to:
  - Dispatch, install, configure, interconnect boxes at remote engines
  - Monitor remote engine status, subscribe to remote events
  - Integration with Virtual Active Network (VAN)



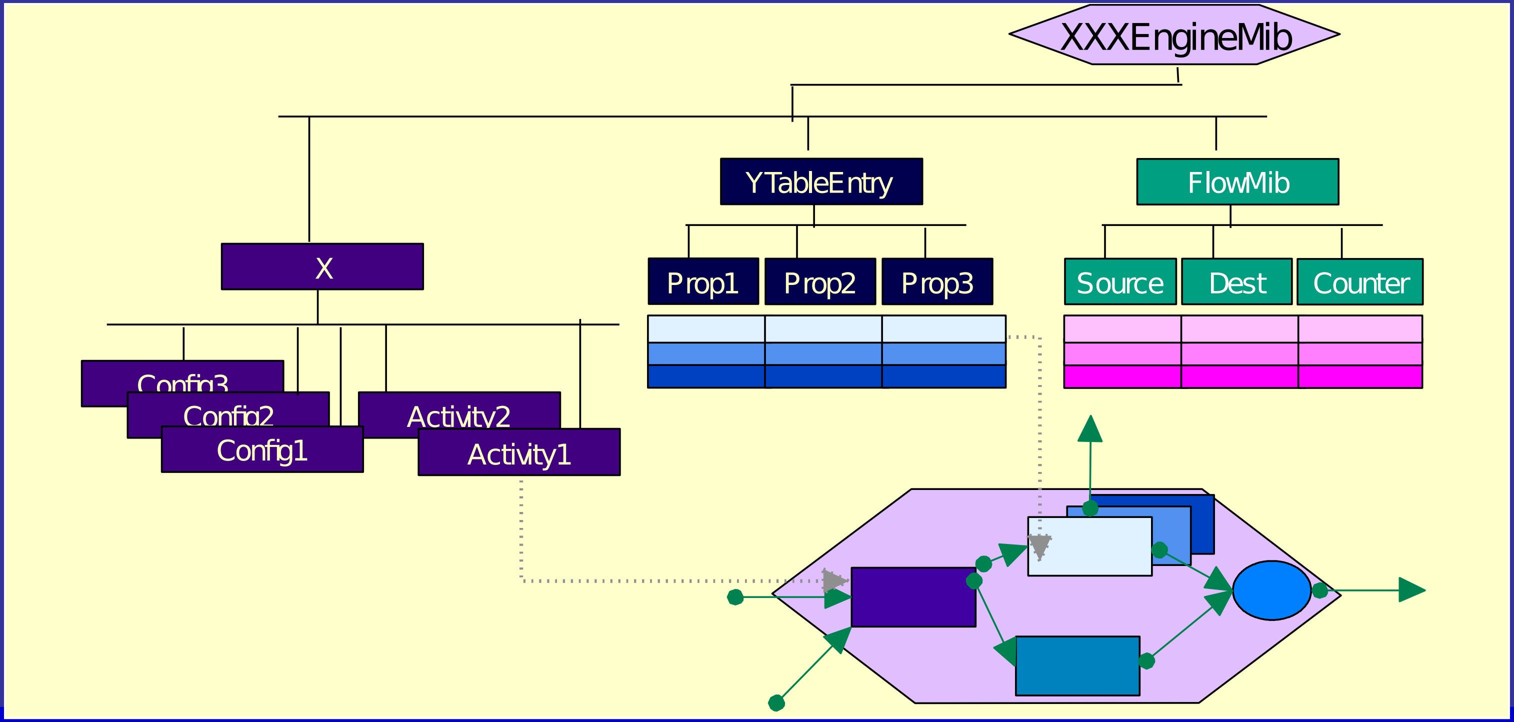
The Challenge of Active Nets Mgmt

- Active components change elements dynamically
- Instrumntn & MIBs must be deployed dynamically
- Mgmt of active components must be automated



Towards Compiler-Generated Mgmt

- Goal: systemic design-time manageability
- Managed properties are integrally designed
- Compiler-generated instrumentation MIBs
- A universal MIB structure unifies semantics



Why a New Language?

- Enable significant domain-specific capabilities
  - Computations over flows
- Simplify programming active nets
  - High-level abstractions of flow processing; End-end composition & coordination
- Compiler-generated support of key functions
  - Manageability [security, resource allocation]
  - Optimization
- Map to heterogeneous node architectures
  - From JVM to ASIC/FPLA...

Status

- Language is available & deployed in ABONE
- Broad applications experiments
  - Active firewalls, routers, IP telephony, QoS control....
- Industry collaborations: Bay,Telcordia, Pentacom...
- Short term goals
  - Complete tooling; VAN; mgmt tools; expand applications base
- Longer term goals
  - Transfer to industry: integrate with routers/switches
  - Develop major applications
  - Automated mgmt of active networks

Synopsis

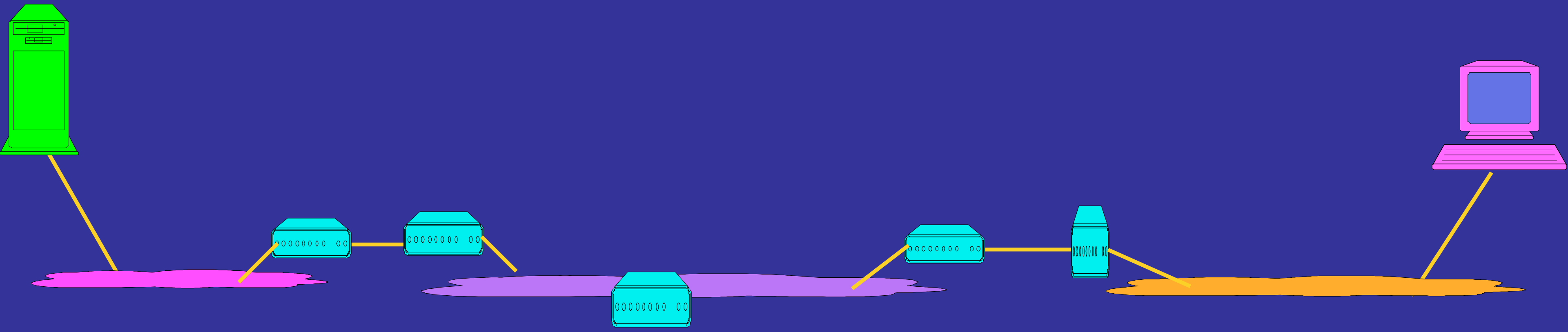
Applications Studies

Active Sensor Networks

- Goal: programmable sensor nets
  - Dynamic adaptation of tasks to observed data
  - Dynamic deployment of processing functions
  - Dynamic resource allocation & QoS management
- The plan
  - Collaborations: CS, Lamont-Doherty, John Hopkins
  - Seismic sensor networks
  - Active application layer, transport layer (QoS), net layer
  - NetScript based

[:) Unfunded] Apps

- Active Global Fence
  - Key idea: enable dynamic fencing of attack sources
  - How: reroute attack traffic through a trapping VAN
- Active Protocol-Based Simulations
  - Key idea: compose simulations using reactive protocols
  - How: extend NetScript boxes with simulation support



?Dimensioning ANet Apps?

- What characterizes applications opportunities
- Dynamic changes/distribution of functions
  - Respond to changes in data
  - Respond to changes in user/traffic needs
  - Respond to changes in network resources availability
- Multi-layer integration of functions